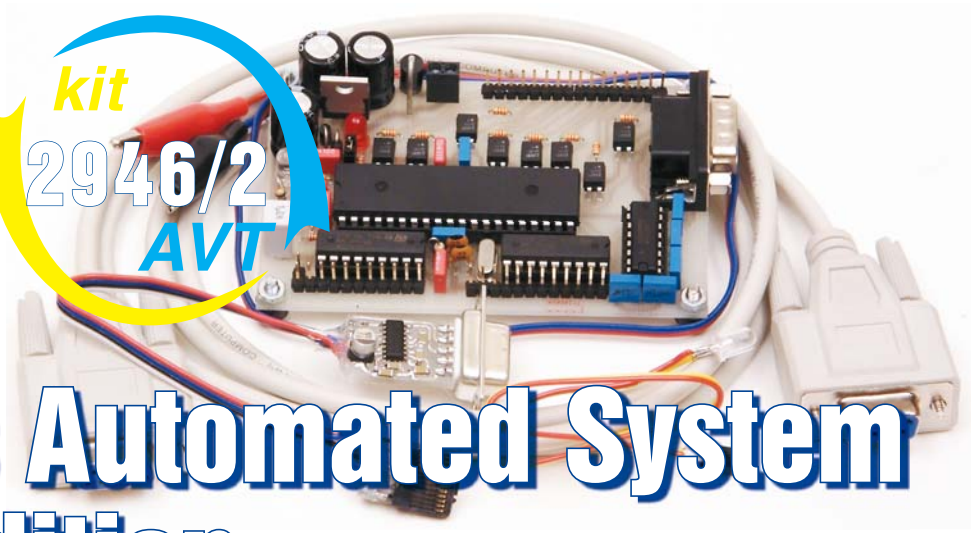


kit 2946/2 AVT

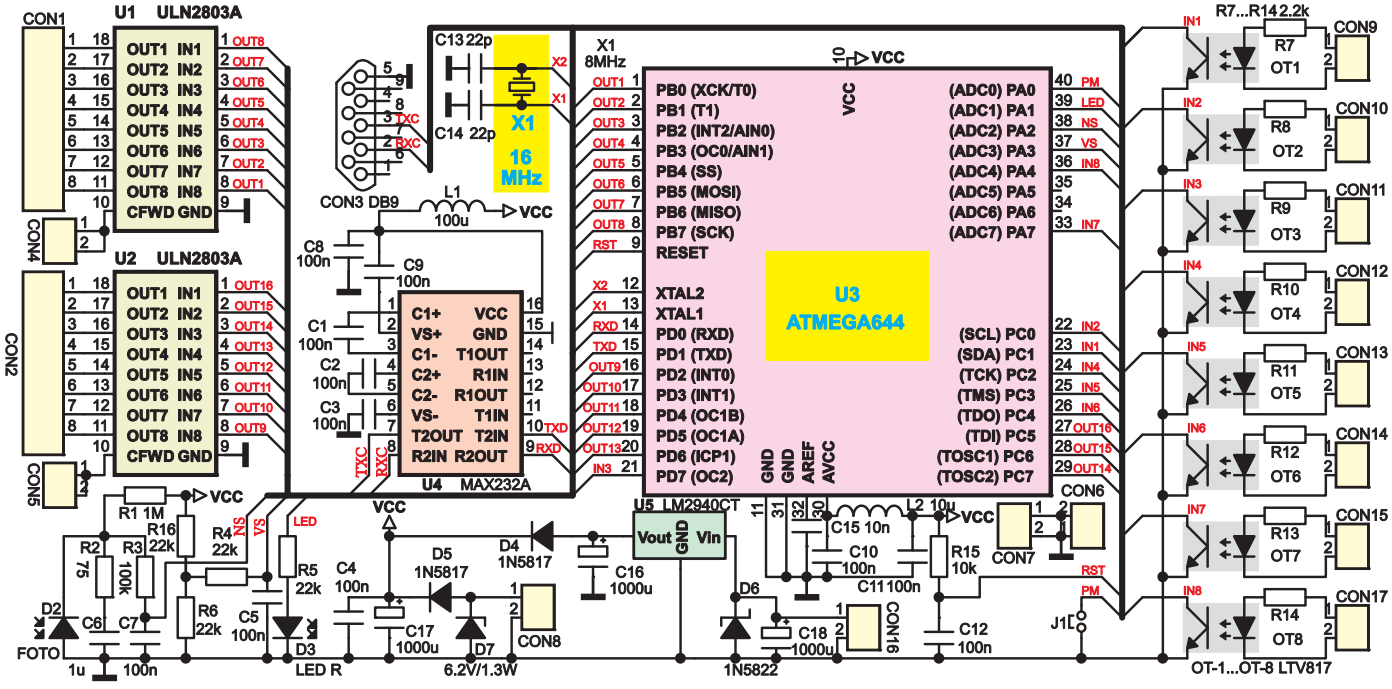
HAS SE – House's Automated System Second Edition

Urządzenie pozwala na tworzenie i łączenie elementów logicznych w sposób programowy. Dzięki takiemu podejściu uzyskujemy maksymalną elastyczność – układ, który ma wejścia i wyjścia, a w środku logikę użytkownika.

Prezentowane urządzenie znajdzie zastosowanie w automatyce domowej oraz jako układ edukacyjny. System pozwala realizować różnorodne zadania sterownicze, przy zachowaniu niezmiennej części sprzętowej.

HAS SE jest rozwinięciem systemu **HAS** prezentowanego już na łamach „Elektroniki dla Wszystkich" (7/2010 str. 21, kit AVT-2946 – PDF z kompletnym artykułem jest dostępny w Elportalu wśród materiałów dodatkowych do bieżącego numeru EdW). Nowy HAS jest odpowiedzią na liczne e-maile. Od tamtego czasu wiele się zmieniło, dodane zostały nowe polecenia, zmienione zostały poprzednie, zmianie uległa nawet koncepcja HAS. Nowa odsłona systemu to programowa implementacja elementów logicznych. Tak, powtórzę jeszcze raz, nowy HAS pozwala na tworzenie i łączenie elementów logicznych w sposób programowy. Dzięki wbudowanym funkcjom, wywoływanym z wiersza polecenia, mamy możliwość definiowania elementów logicznych oraz, co najważniejsze, łączenia ich z wejściami i wyjściami. HAS definiuje zbiór elementów logicznych, takich jak: bramki, przerzutniki, timery, liczniki, opóźnienia itp. Użytkownik ma możliwość wprowadzenia schematu logicznego do pamięci HAS. Dzięki takiemu podejściu uzyskujemy maksymalną elastyczność, układ który ma wejścia i wyjścia, a w środku logikę użytkownika, pozwalającą w dowolny sposób przetwarzać dane. Wprowadzanie potrzebnej, własnej konfiguracji jest bardzo proste i sprowadza się do wpisywania definicji elementów oraz połączeń elementów z wiersza polecenia, a my tylko możemy patrzeć, jak urządzenie ożywa. Dodatkowo nowy HAS zachowuje poprzednie funkcjonalności, czyli możliwość pracy z telefonem komórkowym, co pozwala na zdalną administrację systemu. HAS oferuje 40 poleceń i definiuje 21 typów elementów logicznych. Systemowy cykl pracy w nowym HAS został skrócony 100 razy. Uwaga: nowy HAS bazuje na platformie sprzętowej poprzednika. Posiadacze pierwszej wersji mogą ją łatwo zmodyfikować. Jedynymi zmianami są procesor (ATmega32 została zastąpiona przez ATmega644), rezonator kwarcowy oraz oprogramowanie. Na schemacie ideowym, przypomnianym z tamtego artykułu (**rysunek 0**),

Rys. 0



zmienione elementy są wyróżnione żółtymi podkładami i niebieskim tekstem.

Posiadacze pierwszej wersji układu (kitu AVT-2946) mogą we własnym zakresie wymienić te elementy i wgrać nowy program. Natomiast nowy zestaw AVT-2946/2 zawiera tę samą płytkę, ale nowe elementy według nowego wykazu i rysunku 0.

Możliwości systemu

Funkcje oprogramowania. W HAS operujemy na wejściach i wyjściach. Każde wyjście może być określonego typu, każde wyjście jest elementem logicznym. Od liczby wyjść zależy, ile elementów logicznych możemy zdefiniować. To, jakiego typu elementem logicznym jest dane wyjście, zależy od polecenia wydawanego z wiersza poleceń. Każdy element logiczny może mieć tylko jedno wyjście i zero lub więcej wejść. Wyjątkiem jest licznik, który ma kilka wyjść (alokuje kilka wyjść, co nie znaczy, że jedno wyjście może mieć kilka wyjść). Na wejścia elementów logicznych można podać stany innych wyjść w sposób programowy. Istnieją elementy logiczne, których zadaniem jest łączenie wejść fizycznych z wyjściami, dlatego istnieje możliwość połączenia wejść. Wszystkich wyjść jest w sumie 64 i tyle maksymalnie możemy zdefiniować elementów logicznych. Pierwsze 16 numerów odpowiada wyjściom fizycznym. Wejść jest 16, w tym 8 pierwszych numerów odpowiada wejściom fizycznym.

Dostępne definicje wyjść, elementy logiczne:

- Włączenie lub wyłączenie na stałe, opcjonalnie przyjęcie stanu wejścia.
- Timer, odliczanie. Włączenie lub wyłączanie wyjścia na określony czas, możliwość wywołania funkcji za pomocą wejścia, możliwość zdefiniowania wejścia resetującego odliczanie.
- Cykliczne włączanie i wyłączanie. Możliwość określenia czasu włączenia i wyłączenia, jednorazowo lub w zapętleniu, istnieje możliwość powiązania tej funkcji ze stanem wejścia, opcjonalnie możliwe jest zdefiniowanie wejścia resetującego.
- Tygodniowy harmonogram. Możliwość określenia godziny włączenia i wyłączenia, i wybrania dni, w których to obowiązuje.
- Skok. Możliwość połączenia wejścia z wyjściem, dodatkowo opcja negacji. Dzięki tej operacji wejście staje się wyjściem, co umożliwia jego połączenie do innych elementów.
- Przerzutnik D, Przerzutnik T. Ustawianie stanu wyjścia w oparciu o zmiany stanu wejścia, definicja wejścia wyzwalającego, ustawiającego.
- Przerzutnik JK. Realizuje funkcjonalność przerzutnika typu JK. Posiada 5 wejść.
- Limit. Po określonej liczbie zmian stanu wejścia, następuje wyłączenie wyjścia, przywracanie za pomocą wejścia reset.
- Noc. Wyjście sterowane czujnikiem zmierzchowym.
- OR. Suma logiczna ze zdefiniowanych wejść, możliwość negacji wybranych wejść.
- AND. Iloczyn logiczny ze zdefiniowanych wejść, możliwość negacji wybranych wejść.
- Xor. Alternatywa wykluczająca ze zdefiniowanych wejść, możliwość negacji wybranych wejść.
- Impuls. W zależności od definicji, po pojawieniu się zbocza zarastającego lub opadającego na wejściu powoduje impuls na wyjściu.
- Negacja. Na wyjściu otrzymywany jest stan przeciwny do wejścia.
- Filtr. Stan wejścia musi utrzymywać się przez określony czas, aby na wyjściu pojawił się stan stabilny.
- Dzielnik. Po określonej liczbie impulsów na wejściu następuje odwrócenie stanu wyjścia, opcjonalnie możliwość definicji wejścia resetującego.
- Połączenie wyjścia do wejścia, dzięki temu możliwe jest otrzymywanie zdarzeń z wejścia.
- Licznik. Możliwość definicji licznika o określonej liczbie wyjść, definicja wejścia wyzwalającego oraz resetującego. Licznik liczy w systemie dziesiętnym, po każdym impulsie na wejściu uruchamiane jest kolejne wyjście.
- Przerzutnik RS, wyjście realizuje funkcje przerzutnika RS, definicja wejścia ustawiającego i zerującego.
- Opóźnienie. Stan wyjścia jest opóźniony względem stanu wejścia o zadany czas.
- Podtrzymanie, stan wyjścia będzie podtrzymany o zadany czas po zmianie stanu wejścia.
- Otrzymanie informacji o aktualnie zdefiniowanych elementach na danych wyjściach.

Sterowanie wejściami:

- Sprawdzanie stanu poszczególnych wejść.
- Możliwość ustawienia powiadomienia o zajściu zdarzenia na poszczególnych wejściach. Możliwe warianty to: zbocze narastające, opadające lub oba.

Zarządzanie użytkownikami:

- Dodawanie nowych i nadawanie im początkowych uprawnień.
- Modyfikowanie uprawnień istniejących.
- Usuwanie.
- Wyświetlenie istniejących użytkowników i ich uprawnień.

Administracja systemem:

- Logowanie za pomocą hasła.
- Wylogowanie.
- Zmiana hasła.
- Ustawianie godziny i dnia tygodnia.
- Wyświetlanie godziny i dnia tygodnia.
- Wyświetlanie pomocy.
- Przełączenie do trybu automatycznego (praca z telefonem).
- Przełączenie do trybu manualnego (praca w konsoli).
- Wyświetlanie wersji.
- Wyświetlanie informacji o systemie.

Cechy sprzętu nie uległy zmianie i zostały opisane w poprzedniej wersji HAS. W HAS stany wyjść są buforowane, dlatego wyjścia podłączone do wejść elementów logicznych są opóźnione o jeden cykl, w stosunku do wyjść elementów logicznych. Cykl w HAS wynosi 10ms, okazało się, że jest to wartość możliwie najmniejsza. Jeden element logiczny znacznie obciąża procesor, jest nim... **xor**. Zdefiniowanie 64 takich bramek spowoduje, że HAS zacznie działać wolniej, cykl się wydłuży, przez co czasy (timer, countdown, cycle, keep, delay) przestaną być dokładne. Jest to sytuacja ekstremalna; zdefiniowanie większej liczby innych elementów nie spowoduje wydłużenia cyklu.

Obsługa urządzenia

W tej sekcji zostaną omówione polecenia HAS. Większość poleceń uległa zmianie, więc będzie konieczność omówienia ich od początku. Zacznijmy od funkcji **on** i **off**. Funkcja **on** włącza wyjście na stałe, a funkcja **off** wyłącza na stałe. Możemy więc mówić o logicznym podciągnięciu wyjścia do stanu jedynki lub zera. Do poleceń tych dodano istotne rozszerzenie, mianowicie stan wyjścia może zależeć od jego wejścia. Jest to prosta zależność: wejście jest bezpośrednio połączone z wyjściem. Może to wydawać się nieco zawiłe, ale do tego wejścia może być podłączone inne wyjście. W parametrze tego polecenia podaje się wyjście, składnia: *[on|off] <numer_wyjścia> [if out <numer_wyjścia_dołączonego>]*. Gdzie *<numer_wyjścia>* jest to numer wyjścia, do którego chcemy przypisać daną funkcję, natomiast *<numer_wyjścia_dołączonego>* jest to numer wyjścia, które jest połączone z tym wyjściem (inaczej mówiąc, jest podłączone do jego wejścia, jeszcze inaczej mówiąc: steruje tym wyjściem). Przykład poleceń widoczny jest na **rysunku 1**. Kolejne dwa polecenia: **timer** i **countdown**, uruchamiają funkcję powodującą odpowiednio wyłączenie i włączenie wyjścia na zadany czas. Bez dodatkowych przełączeń, funkcje te wykonują się od razu, istnieje też możliwość warunkowego wywołania

```
> on 2,3,5-9
OK
> off 1-16
OK
> on 1 if out 2
OK
> off 3 if out 4
OK
```

Rys. 1

Rys. 2

```
> timer 1 on 0:00:00:10:00
OK
> countdown 2 on 0:00:00:05:00
OK
> timer 1 on 0:00:00:10:00 on out 3
OK
> timer 1 on 0:00:00:10:00 on out 3 repeat reset 4
OK
```

odliczenia zadanego czasu, a tym samym zmiany stanu wyjścia. Warunkowe wyzwalanie czasu zależne jest od stanu wejścia elementu, czyli wyjścia dołączonego do tego wejścia: odliczanie czasu następuje po pojawieniu się stanu wysokiego na tym wejściu. Po zmianie stanu tego wejścia czas może zostać odliczony jednorazowo lub wielokrotnie. Dodatkowo istnieje możliwość definicji wejścia resetującego, wtedy stan wysoki na tym wejściu powoduje resetowanie czasu, tym samym powrócenie do stanu sprzed odliczania czasu: jedynki dla funkcji **timer**, i zera dla funkcji **countdown**. Odliczanie czasu jest synchroniczne w stosunku do stanu wejścia. Po wyzwoleniu czasu, stan wyjścia może powrócić do zera, jednak nie przerwie to odliczania. Wejście resetujące działa asynchronicznie i pojawienie się stanu wysokiego powoduje natychmiastowe przerwanie odliczania. Przykład widoczny jest na **rysunku 2**, gdzie należy zwrócić uwagę, że czas podawany jest w nowym formacie. Pola definicji czasu rozdzielone są dwukropkami, znaczenie pól to odpowiednio: ***dzień, godzina, minuta, sekunda***, ***milisekunda pomnożona przez 10***. Polecenie **cycle** pozwala na zadanie czasu wyłączenia i włączenia wyjścia. Jeśli polecenie zostanie wydane, początkowo wyjście zostanie wyłączone na zadany czas, po czym włączone na zadany czas, i znowu wyłączone. Cykl może zostać zapętlony przełączeniem **repeat**, dzięki temu uzyskujemy coś w rodzaju generatora o zmiennym wypełnieniu. Dodatkowo istnieje możliwość uruchomienie tego cyklu w oparciu o stan wejścia. Podobnie jak w przypadku poprzedniego polecenia, wyzwalanie cyklu za pomocą wejścia odbywa się zboczem narastającym i jest synchroniczne, czyli jeśli stan na wejściu wyzwalającym zmieni się na zero, cykl nie zostanie przerwany. Ponadto możemy zdefiniować i dołączyć wyjście resetujące cykl. Dołączenie innego cyklu do wejścia wyzwalającego pozwala na uzyskanie bardziej złożonych przebiegów na wyjściu – **rysunek 3**. Polecenie **daily** nie uległo zmianie, pozwala ono na włączanie wyjścia w zależności od zaplanowanego harmonogramu. W argumentach polecenia podaje się czas załączania i wyłączania (godzinę, minutę, sekundę) oraz dni tygodnia, w które plan obowiązuje. By uzyskać bardziej skomplikowany harmonogram, np. dwa okresy w ciągu dnia, trzeba skorzystać z funkcji **OR**, omawianej w dalszej części artykułu – **rysunek 4**. Polecenie **jump** jest bardzo istotne, pozwala ono na połączenie wejścia fizycznego do wyjścia, dzięki temu możemy określić stany wejść. Podłączenie wejścia do wyjścia umożliwia następnie dołączenie tego wyjścia do wejść układów logicznych. Trzeba jednak podkreślić, że takie przejście marnuje wprawdzie jedno wyjście, jednak wiele upraszcza w implementacji wielu funkcji. Dodatkowo w poleceniu można podać, że stan wejścia ma być negowany. Polecenia **d** oraz **t** definiują odpowiednio element przerzutnika D oraz T. Przerzutniki te mają wejście ***clk*** oraz wejście ***d*** lub ***t*** (w przypadku przerzutnika T), do których można dołączyć sygnał z innych wyjść. Istnieje jeszcze polecenie **JK**, definiujące przerzutnik JK. Polecenie to występuje w dwóch wariantach, jeden bez definicji wejść RS, drugi z definicją wejść RS. Działanie elementu **limit** jest nieco bardziej złożone: w argumencie polecenia definiującego ten element podaje się liczbę (*amount*) cykli, przez jaką wyjście ma być aktywne. Początkowo na wyjściu elementu utrzymuje się stan wysoki, po liczbie zboczy narastających na wejściu *clk* odpowiadającej zmiennej *amount* następuje zmiana stanu na niski. Zerowanie, przywrócenie do stanu początkowego odbywa się poprzez wejście *reset*. Polecenie **nigth** definiuje element czujnika oświetlenia. Na wyjściu tego elementu utrzymuje się stan wysoki przy braku oświetlenia, i niski przy oświetleniu. Dodatkowo przełączeniem ***reverse*** można zanegować stan wyjścia. Polecenia **and**, **or**, **xor** definiują odpowiednio element bramki logicznej AND, OR, XOR. Liczba aktywnych wejść bramki odpowiada liczbie wyjść podanej na liście argumentu *out*. Do wybranych wejść bramki po prostu można podłączyć wybrane wyjścia, a każda bramka ma 64 wejścia. Co ważne, w opcjonalnym argumencie *neg* podaje się listę zanegowanych wejść bramki, dzięki temu możemy uzyskać bramkę NAND i NOR. Polecenie **impulse** definiuje element impuls. Element ten ma jedno wejście *out*, do którego dołącza się wyjście. Na wyjściu tego elementu pojawia się impuls o czasie trwania jednego cyklu HAS, przy pojawieniu się zbocza narastającego lub opadającego na wejściu, w zależności od konfiguracji. Plecenie **not** definiuje element inwertera: na wyjściu elementu pojawia się stan przeciwny do stanu wejścia. Element ten ma jedno wejście *out*. Polecenie **filter** definiuje element filtra. Element ten ma jedno wejście, stan wysoki na wejściu musi utrzymywać się przez podaną w parametrze *value* liczbę cykli, aby na wyjściu pojawił się stan wysoki. Każda zmiana stanu z wysokiego na niski zeruje wewnętrzny licznik i odliczanie rozpoczyna się od początku. Polecenie **div** definiuje element dzielnika. Element ten ma jedno wejście zegarowe *out* oraz wejście *reset*, które może być opcjonalnie połączone. Po podanej w parametrze ***by*** liczbie zboczy narastających na wejściu *out*, następuje odwrócenie stanu wyjścia. Polecenie **join** definiuje element łączący wyjścia z wejściami. Element definiowany jest w obrębie wejść, a nie wyjść, jak było do tej pory. Dzięki temu elementowi istnieje możliwość doprowadzenia stanu wyjść na wejścia, co może być wykorzystane do generowania zdarzeń lub odczytania stanów wyjść, np. za pomocą SMS (stany wyjść można odczytać poleceniem **describe** wydawanym z konsoli). Polecenie **counter** definiuje licznik. W odróżnieniu od innych elementów, licznik zajmuje kilka wyjść, poczynając od wyjścia podanego w pierwszym parametrze. Definiując licznik, określamy, ile ma mieć wyjść (*outputs amount*). Licznik ma wejście *clk*, na które trzeba podać sygnał wyzwalający

Rys. 3

```
> cycle 3,4 on 0:00:00:00:10 off 0:00:00:00:20
OK
> cycle 3,4 on 0:00:00:00:10 off 0:00:00:00:20 repeat
OK
> cycle 3,4 on 0:00:00:00:10 off 0:00:00:00:20 repeat on out 1
OK
> cycle 3,4 on 0:00:00:00:10 off 0:00:00:00:20 repeat on out 1 reset 2
OK
```

Rys. 4

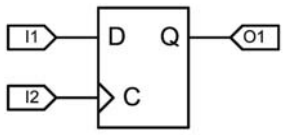
```
> daily 64 on 13:00:00 off 15:00:00 day 1-7
OK
> daily 63 on 22:00:00 off 6:00:00 day 1-7
OK
> or 1 out 64,63
```

Rys. 5

```
> jump 62 in 1
OK
> jump 61 in 2
OK
> jump 60 in 3
OK
> d 1 clk 62 d 61
OK
> t 2 clk 62 t 61
OK
> jk 3 clk 62 j 63 k 64
OK
> jk 3 clk 62 j 63 k 64 r 1 s 2
OK
> limit 4 amount 3 clk 62 reset 63
OK
> night 5
OK
> night 5 reverse
OK
> xor 4 out 62-64
OK
> or 5 out 62-64
OK
> and 6 out 62-64
OK
> impulse 7 out 64 on rise
OK
> impulse 7 out 64 on fail
OK
> not 8 out 64
OK
> filter 9 value 10 out 64
OK
> div 10 by 3 out 64
OK
> div 10 by 3 out 64 reset 63
OK
> join 8 out 1
OK
> join 8 off
OK
> counter 9 outputs amount 3 clk 64
OK
> rs 12 set 64 reset 63
OK
> delay 13 out 64 time 0:00:00:00:05
OK
> keep 14 out 64 time 0:00:00:10:00
OK
```

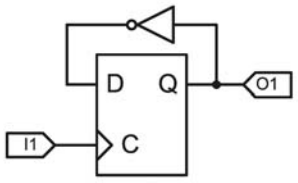
kolejne przełączenie, przełączanie następuje na zboczu narastającym. Opcjonalnie można określić wyjście resetujące. Licznik maksymalnie może mieć 16 wyjść. Kolejne polecenie **rs** definiuje element przerzutnika typu RS. Element ten ma dwa wejścia, jedno ***set***, drugie ***reset***. Podanie stanu wysokiego na wejście ***set*** ustawia wyjście w stan wysoki, podanie stanu wysokiego na ***reset*** ustawia wyjście w stan niski. Polecenie **delay** definiuje element opóźnienia. Stan na wyjściu jest opóźniony w stosunku do stanu wejścia o zadany czas. Ostatnie polecenie **keep** definiuje element podtrzymania. Element ten ma jedno wejście, na którego podanie stanu wysokiego powoduje ten stan na wyjściu, po ustąpieniu stanu wysokiego na wejściu, na wyjściu utrzymuje się stan wysoki przez czas podany w parametrze polecenia. Przykłady poleceń widoczne są na **rysunku 5**. Dodano również polecenie **info**, wyświetlające informacje o systemie, oraz polecenie **version**, zwracające aktualną wersję firmware'u. Lista wszystkich poleceń wraz z dozwolonymi parametrami dostępna jest w załączniku *diagramy syntaktyczne HAS*.

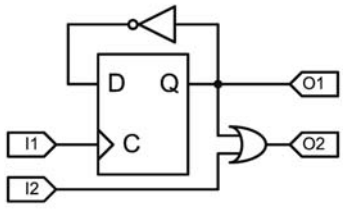
Przejdźmy teraz do przykładów. Przykłady będą proste, jednak będą obrazowały zamysł twórcy. We wszystkich przykładach będziemy wykorzystywali przerzutnik typu D. Pierwszy przykład obrazuje działanie przerzutnika D – **rysunek 6**. W przykładzie wykorzystamy dwa wejścia oraz jedno wyjście. Wejście 1 będzie dołączone do wejścia D przerzutnika, wejście 2 do wejścia CLK. Wyjście przerzutnika dołączymy do wyjścia 1. Początkowo musimy zbudować most, a więc doprowadzić stany wejść do wyjść. Robimy to następującymi poleceniami: ***jump 64 in 1***, ***jump 63 in 2*** (w konsoli po każdym poleceniu naciskamy E*nter*). Następnie tworzymy przerzutnik na wyjściu 1: ***d 1 clk 63 d 64***. Jak widać, do wejścia *clk* został poprowadzony sygnał z wyjścia 63, do którego przekazywany jest stan z wejścia 2. To tyle. Możemy teraz przetestować, czy przerzutnik działa. Osoby niewiedzące, jak działa przerzutnik D, mogą przeprowadzić doświadczenia.



Rys. 6

Rys. 7





Rys. 8

W kolejnym przykładzie zapętlimy zanegowane wyjście przerzutnika z wejściem D, uzyskując asynchroniczny przerzutnik typu T – **rysunek 7**. W przykładzie wykorzystamy jedno wejście oraz jedno wyjście. Sygnał z wejścia 1 zostanie doprowadzony do wejścia CLK przerzutnika. Sygnał z wyjścia Q do wyjścia 1. Ponieważ przerzutnik nie ma wyjścia zanegowanego, będzie nam potrzebny element inwertera. A więc po kolei:
jump 64 in 1, ***d 1 clk 64 d 63***, ***not 63 out 1***.
I to wszystko! Działa!

W ostatnim przykładzie wzbogacimy przykład poprzedni dodatkową bramką OR – **rysunek 8**. Wyjście bramki OR dołączymy do wyjścia 1, do jednego z wejść doprowadzamy sygnał z wyjścia Q przerzutnika, do drugiego sygnał z wejścia 2. Po wydaniu poleceń z przykładu 2 wydajemy jeszcze polecenia: ***jump 62 in 2***, ***or 2 out 1,62***.

Za pomocą SMS mamy dostęp do ograniczonej liczby poleceń, dostępne są następujące polecenia: **on**, **off**, **timer**, **countdown**, **daily**, **cycle**. Lista poleceń jest celowo ograniczona, z uwagi na fakt, że przez SMS powinny być możliwe tylko polecenia bezpośrednio związane ze sterowaniem wyjść. W celu uchronienia zdefiniowanej logiki przed uszkodzeniem przez polecenia wydawane przez SMS, trzeba odpowiednio nadać uprawnienia do poszczególnych wyjść, z poziomu poleceń administrujących użytkownikami.

Montaż i uruchomienie

Montaż należy przeprowadzić zgodnie z instrukcjami podanymi w poprzedniej wersji HAS (przypominam, że plik PDF z artykułem dostępny jest w Elportalu wśród materiałów dodatkowych do tego numeru). Jedyną różnicą jest programowanie procesora, a zwłaszcza jego fusebitów. UWAGA, aby program działał poprawnie, niezbędne jest zaprogramowanie fusebitów, pamięci EEPROM oraz pamięci FLASH procesora. Oczywiście niezbędne pliki, w tym szczegółową instrukcję programowania można znaleźć w Elportalu.

Szymon Janek
sj@post.pl

Wykaz elementów

R1	1MΩ/0,125W
R2	75Ω/0,125W
R3	100kΩ/0,125W
R4-R6,R16	22kΩ/0,125W
R7-R14	2,2kΩ/0,125W
R15	10kΩ/0,125W
C1-C5,C7-C12	100nF/63V/MKT
C6	1μF/63V/MKT
C13,C14	22pF
C15	10nF/63V/MKT
C16-C18	1000μF/16V
D2	fotodioda
D3	LED R/5mm
D4,D5	1N5817
D6	1N5822
D7	Zenera 6,2V/1,3W
U1,U2	ULN2803A
U3	ATmega644-20PU
U4	MAX232A
U5	LM2940CT-5.0
OT1-OT8	LTV817
X1	16MHz/4mm
L1	dławik 100uH
L2	dławik 10uH
CON1,CON2,CON4-CON15, CON17,J1	
łącze szpilkowe proste „goldpin" 1*40pin + 1*4pin	do łamania
CON3	wtyk D-SUB kątowy do druku L=7,2mm (to z bolcami)
CON16	ARK2/SM
podstawka standardowa 18pin 0,3" 2szt.	
podstawka standardowa 16pin 0,3" 1szt.	
podstawka standardowa 40pin 0,6" 1szt.	

Komplet podzespołów z płytką jest dostępny w sieci handlowej AVT jako kit szkolny AVT-2946/2.